Building High-Performance, Scalable Applications

Filo D'Souza
Program Manager
SQL Server
Microsoft Corporation

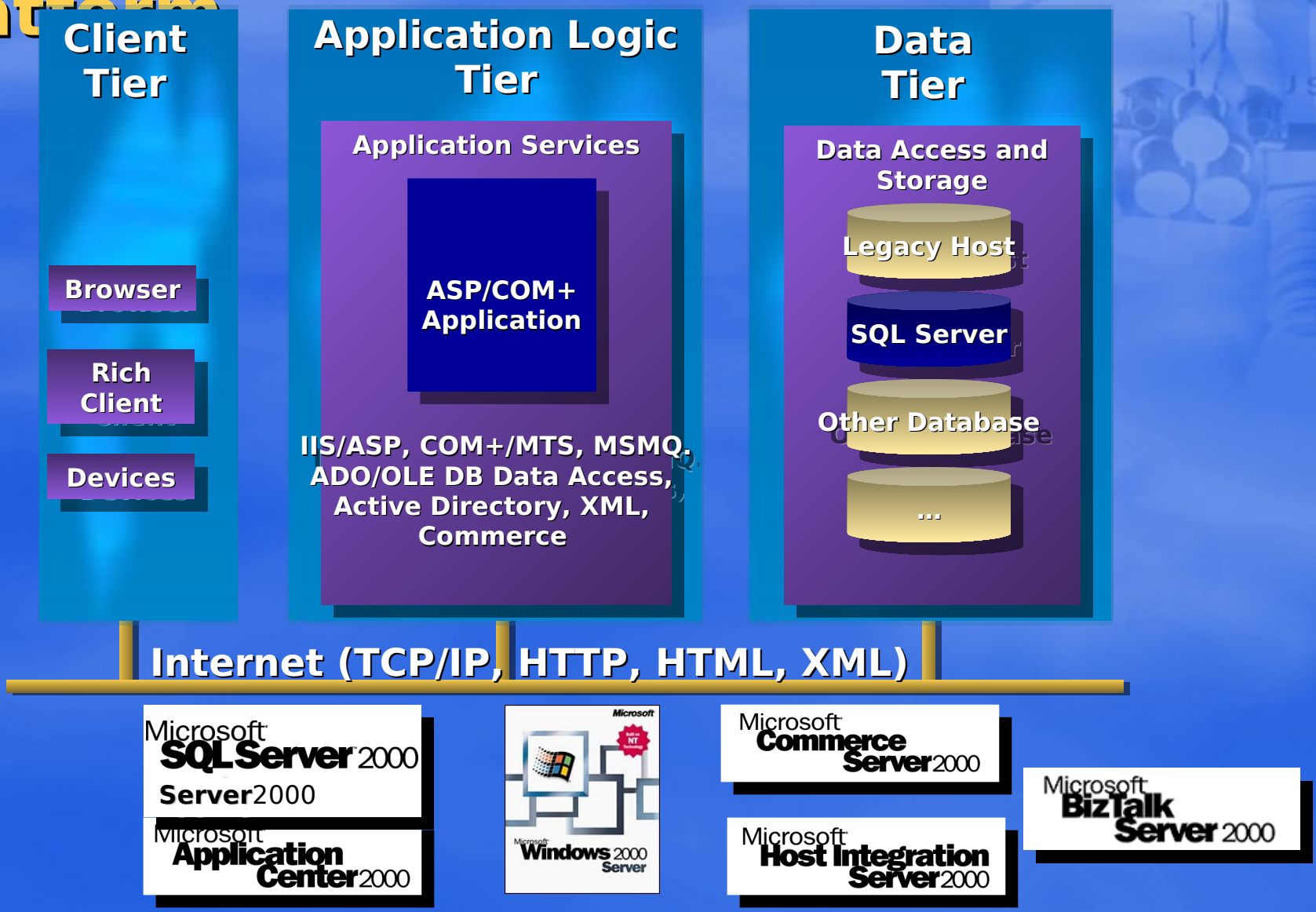

POWER UP
Windows DNA 2000
Readiness Conference
featuring SQL Server 2000

Objectives

- SQL Server™ performance and scalability from an application development perspective
- Key performance characteristics

Windows DNA 2000

Next Generation Web Application Platform



SQL Server Performance

- Self configuring, managing, tuning
  - Dynamic memory/lock management
  - Auto stats creation/update
  - Fully integrated intra-query parallelism
  - Automated index tuning
  - Intelligent I/O
  - More…

Agenda

- Plan caching
- Cursors
- SQL Server 2000 performance enhancements

Plan Cache

- Optimizer compiles query tree into an execution plan
  - Normalization
  - Optimization (cost-based) uses stats
  - Prune heuristics
- Execution plan – data structure
- Stored in plan cache

Plan Caching

- Ad-hoc workloads
  - Adhoc caching
  - Auto parameterization
- Defined workloads
  - Stored procedures
  - ExecuteSQL
  - Prepare / Execute / Unprepare

Ad-Hoc Plan Caching

- Cache the plan from adhoc statements
- Subsequent statement has the exact exact textual match

  Example:

  Q1: select * from employees where fname like ‘S%’

  Q2: select * from employees where fname like ‘D%’

  Q3: select * from employees where fname like ‘S%’

  - Q3 will used cached plan from Q1
  - Q2 needs to be compiled

Auto-Parameterization

- Only on “simple” statements
- Determines constants that are probably/could be parameters
- Parameterized template is created
- Other queries can use template if “safe”
- Recommendation:
  Do not rely on this for defined workloads

Auto-Parameterization Forms

- INSERT <table> VALUES ({constant} | NULL | DEFAULT}, ..)
- DELETE <table> WHERE <key-exp>
- UPDATE <table> SET <col> = <constant> WHERE <key-exp>
- SELECT <col-list> FROM <table> WHERE <key-exp> ORDER BY <col-list>

Auto-Parameterization

- Safe
  - Template: INT @P, SELECT fname, lname FROM employees WHERE emp_id = @P
- Unsafe
  - SELECT fname, lname FROM employees WHERE (salary + bonus) > 30000
- Optimizer decision

Stored Procedures

- Batch of statements stored within the server
- Persistent object management
- Parameters are specified by the application
- Plans are compiled / cached on execution
  - WITH RECOMPILE option
- Dynamic statements with SP

Stored Procedures

- Business logic closest to data
- Performance
  - Next number generation
  - Minimizes network traffic
  - RPC execution avoids parameter processing and statement parsing
- UDFs

ExecuteSQL

- Created by application
- Does not require persistent object management

Example:

```
Sp_execute('insert TableA values (@p)','@p float',1)
Sp_execute('insert TableA values (@p)','@p float',3)
Sp_execute('insert TableA values (@p)','@p float',1)
```

- Same cache plan will be used
- SQLExecDirect, IcommandWithParameters

Prepare / Execute

- Created by the application
- Text of batch is sent once on Prepare
- A handle will be used to repeatedly Execute
- Cleaned from the DBMS at UnPrepare
- SQLPrepare/SQLExecute, ICommandPrepare

Inappropriate Plan Sharing

- Re-using avoids compilation and optimization
- Optimal execution time
  - Not achieved if not using the optimal plan for given parameter values

Example:

```
Select * from T where c>=5 and c<=10 (index lookup)
Select * from T where c>=5 and c<=100 (table scan)
```

Issue: ExecSQL Versus Prepare / Execute

- Database driver
  - Isolated from the application logic
  - Special casing
  - Passing hints
- Predicting usage by user
- Application server
- N+2 costing

SQL Server 2000 Enhancements

- Reduced roundtrips with Prepare/Execute
  - Types of all parameters specified
  - Result-set meta data is not requested between prepare and execute
  - The number of round-trips for N executions is now N

SQL Server 2000 Enhancements

- **Reduced meta-data overhead**
  - **Selects – meta data for each column**
    - **meta-data is cached on the client-side (prepare/execute)**
  - **Parameter processing/meta-data**
    - **large cost for wide tables (30%)**
    - **Optimization on the SQL Server client**
    - **Only data values and data length is read**

Recommendations

- Stored procedures
- Use Prepare/Execute instead of SQLExecute
  - Share connection
- Parameterization when possible
  - Range of values should not change drastically
  - App should specify types of parameters
- Sharing rules

Cursors – Why?

- Applications - database driver isolation
- Scroll/update within rowset
- Requirement for multiple statements per connection
- Possibility of a small subset of rows used, from requested total
- Other application requirements

Efficient Use Of Cursors

- Subtle changes in cursor types can have a big impart on performance as you scale up
- Choose lowest cost cursor for your requirements
  - Default row set, fast forward only
  - Unless there are functionality requirements from the ‘rich’ cursor models
- Impact on server and n/w resources
- Respect Transactional control (isolation levels)

Default Result Set – Firehose

- ‘Base’ model – single SQL statement
- Full consumption of result set
- Small result set – esp. singletons
- Benefits
  - Minimize server resources
  - ‘Request’ not required
- Issue: connection busy until results consumed

Fast Forward Only

- Optimized forward-only, read-only cursors
- Connection busy minimized – multiple statements
- Thin layer over the query
- Operates directly on base table(s)
- Supports only fetch next
- Reflects committed changes to data

Static Cursors

- Snapshot of answer set is materialized in a temp table at cursor open
- Read only – underlying tables cannot be updated
- All scrolling options are available
- Not ‘dynamic’

Keyset Cursors

- Creates a keyset of the answer rowset in a temp table at cursor open
- All scrolling options are available
- Requires a unique index on every table in the select
- Obtains non-index columns from data pages
- Updates visible
- Membership is fixed – fetch to absolute position

Dynamic Cursors

- Operate directly on base table
- All scroll options available except absolute position
- All changes visible when scrolling
- Limited join techniques
- Downgrade possible

Cursor Downgrade

- Generally when materialization of the answer set on the server is required
  - Depending on original request
    - ORDER BY not covered by an index
    - TOP, GROUP BY, UNION, DISTINCT…
  - Can downgrade to static, keyset or dynamic
  - Subject to change when product is enhanced
- Recommendation: TEST
  - SQL_SUCCESS_WITH_INFO

Static / Keyset

- Static
  - Entire answer set materialized
  - Cheapest subsequent fetch
- Keyset
  - Rids / keys materialized
  - Other columns are fetched
  - Next cheapest fetch
- Asynchronous population
  - Sp_configure ‘cursor threshold’ n

SQL Server 2000 Enhancements - Cursors

- Internal
  - Cursor compile plan
  - Cursor execution plan
  - Fetch next for static and keysets
  - Keyset fetch
  - Cursor foot print (memory)

SQL Server 2000 Enhancements - Singletons

- Selects (general)
  - Eliminating memcopies
  - Speeding column access
- IRow interface (OLE/DB)
  - Direct access to columns of a single row object
  - Forward sequential access

General Performance

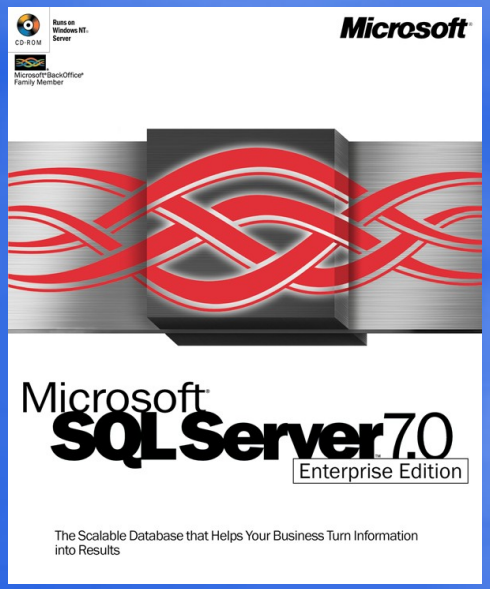
- Processing ‘Default’ parameters
  - Flag from first/previous insert
- Speeding up batch processing (inserts and selects)
  - Combine inserts
- Shared memory netlib (NT)
- SAN technology

Connection Pooling

- Reuse client connection to DB server
- Performance gains from reduced network traffic/latency and server CPU
- Driver determines when a connection should be dropped
- Pool behavior

SQL Server today…

- CRM
- ERP
- E-Commerce
- Data Warehousing



PEOPLESOFT
BAAN
SAP
CLARIFY
GREAT PLAINS
SOLOMON SOFTWARE
SIEBEL
JDEDWARDS ENTERPRISE SOFTWARE
INFORMATICA
LAWSON SOFTWARE
VISUAL COMMERCE CONSTRUCTOR
ONYX SOFTWARE
SAGENT
KNOSYS
PIVOTAL
COGNOS BETTER DECISIONS EVERY DAY

Scaling Considerations

- Server resources – many concurrent users
- Dynamic requirements
- Processing, cleanup
- Network roundtrips

Summary

- Ensure strong plan caching
  - Stored procedures
  - Prepare/Execute
  - Parameterize
- Cursors
  - Use cost effective cursors
  - Test for downgrade
- Many optimizations in SQL Server 2000 - automatically

More Information

- Books on Line
- Beta2 News groups

  Microsoft.beta.Shiloh.*
- SQL Server 7.0 Home page

  http://www.microsoft.com/sql/
- Microsoft Product Support Services

  http://search.support.microsoft.com

Questions?

Filo D’Souza
(filods@microsoft.com)

POWER UP

Microsoft